

Landlords: A Multiplayer Monopolization Game for the Android OS

By Tyler Kerry
100740419

Honours Project (COMP 4905)
School of Computer Science
Carleton University

Supervised by Dr. Tony White
School of Computer Science
Winter 2011

Abstract

Mobile phones may have existed for several decades now, but never have they provided us with the technology they provide today. This project intends to examine the pros and cons of developing for the Android OS, while attempting to build a distributed client-server architecture, as well as an outline-tracing solution to locate buildings on Google Maps. It will make use of many technologies such as Android, Google Maps, GPS, JSON, Restlet, and SQLite. The client and server model demonstrates considerable success in a first attempt at the Android architecture, and the outline-tracing algorithm demonstrated here has been shown to be very successful. We will discuss the software design, choices, technologies, and results of this project while gaining insight into developing for the mobile world.

Acknowledgements

First of all I would like to thank Dr. Tony white for supervising this project and for the support he has supplied. I would also like to acknowledge Google and the Android Team for providing the tools that were necessary to complete this project, including the Android SDK, Android emulator, and Google Maps. Portions of this project and report are reproduced from work created and shared by Google and used according to the terms described in the Creative Commons 3.0 Attribution License. These include the Android Robot, and three tab icons used in the user interface of the Android application. All of these can be found with the Android software development kit which is freely available.

Contents

1. Introduction	6
a. Background	6
b. Motivation.....	7
c. Goals.....	8
2. Description.....	9
3. Methodology.....	12
a. Server.....	12
b. Client.....	13
c. Location Tracer.....	15
4. Results	19
5. References.....	20
6. Appendices.....	21
a. CD Content.....	21
b. Instructions	21

Table of Figures

Figure 1: View of the Locations tab.....	9
Figure 2: The Map view	11
Figure 3: Service API.....	13
Figure 4: Typical Google map result.....	16
Figure 5: Map from Figure 4 after being read and traced	17
Figure 6: Pixel exclusion.....	18

1. Introduction

a. Background

Very few people could argue with the fact that as technology dramatically increases, so does our lifestyle become more hectic. Technology has allowed for and encouraged people to be forever on the move, from one place to the next, without ever losing connectivity to our electronic world. Humans are forever on a search for a faster way to accomplish things, whether it's a revolutionary contribution to medical science, the business world, or even a new social networking site that everybody wants to use.

With this desire comes a heavy dependence on mobile technology. From laptops to handheld devices, people are finding new ways to stay connected and stay entertained. In 2010, mobile phone subscriptions numbered 4.6 billion and it was estimated that by the end of that year, they would reach 5 billion.¹ While that includes high-end technological mobile devices such as the iPhone and Android-enabled devices, it also includes many outdated and obsolete devices. However, that is still an outrageous number. The U.S. census provides us with an estimated world population of 6.8 billion people in 2010.² When compared to 4.6 billion mobile phone subscriptions, we can see a ratio of approximately two phones to every three people. This is a staggering number to think that one has contact with 66% of the world at their fingertips.

However, despite this high-paced lifestyle, there is still one thing that humans cannot seem to live without, and that is entertainment. This is part of the motivation to create this project, to provide a source of entertainment to a vast amount of mobile users. This increase in phone technology provides game makers with a valuable resource. People who have a bit of spare time can pick up their phone and spend four or five minutes playing their favourite mobile game, before they are off to continue their busy life. It is a cheap form of entertainment, which is an excellent platform for independent game designers to make a hit without the vast team and funding required to produce a triple-A title.

¹ <http://www.cbsnews.com/stories/2010/02/15/business/main6209772.shtml>

² <http://www.census.gov/ipc/www/idb/worldpopgraph.php>

b. Motivation

An interesting concept in the field of mobile games is augmented reality. Augmented reality is a relatively vague term that is used to reflect the state of a computer simulation based on factors in the real world. Although this term applies to technologies in general, with the technologies provided by mobile phones such as GPS and a camera, the mobile phone can be used to achieve this effect. As such, it is the goal of this project to incorporate GPS and the user's location to change the way that the game is played, and achieve a bit (if albeit small) amount of augmented reality.

A problem that has existed in the field of Computer Science is finding efficient and reliable algorithms for reducing the number of vertices that exist in a polygon. There are a number of applications for these algorithms, such as smoothing grainy images. This problem is one that I was motivated to approach for the benefit of my knowledge, as well as Computer Science as a field.

Another motivation for this project is to attempt to utilize the variety of packages and resources that the Android operating system provides, and combine them together into a structured and meaningful representation. The software development kit for Android is extensive, and contains invaluable resources that can be used to enrich custom applications. Several of these include the Restlet framework, JSON, and the Google Maps API that can be easily integrated into Android applications. Since mobile technology surely shows no sign of slowing down anytime soon, this project is intended to give the author some experience in working with this mobile architecture, and provide the reader with some tips and insight that I was granted while working on this project.

c. Goals

One of the goals of this project is essentially to break it down into three smaller projects that fulfill specific roles when interacting with each other within the architecture of this project. The three projects that will exist will be the server, the client, and the location tracer.

It is the aim of this project to create a server to serve as the central access point for the rest of the project. The server provides resources and methods to allow clients to interact in a meaningful way. To provide some information to the server administrator, it is my intent to create a window for the administrator to see incoming and outgoing requests, as well as monitor the database. The server will be coded using a web framework known as Restlet.³ Restlet provides web services and clients a way with which they can communicate in a RESTful manner. RESTful architecture will be discussed in greater detail later in this report.

The client is another extremely important entity in this architecture. Without the client, the server, and this entire project, has no reason for existing. To provide people access at any time, it is my goal to create the client on the Android SDK and to be able to run it on an Android-enabled mobile device.

The third mini-project, and probably least important yet most interesting, is the location tracer. The location tracer is a piece of software developed by the author to correctly enable location finding services required by the server. It is intended to analyze an image provided by Google maps, and to provide a mapping of vertices that respond to a specific location on that map. This piece of software will require algorithms to decipher this information in an efficient and reliable manner.

³ <http://www.restlet.org/>

2. Description

This software was designed to be an Android game that is heavily based on using Google Maps to provide a visual representation of what essentially becomes a board game. In particular, it is intended to be real-estate management game, where users are given the option to buy and sell property, and make money off of charging rent. The goal to this project is that with a dedicated server running, any person with an Android-enabled phone and the Landlords application can sign-in using an existing username and password, or can register as a new user. Once this is done, they are provided with a set amount of money. From the main screen (after a sign-in), they have access to three tabs. The first tab is the settings tab which provides some basic information (such as the server they are signed into, and how much money they have). The second tab provides a list of the locations that the user currently owns (see Figure 1: View of the Locations tab). This tab also contains an update button which sends a request to the server for their refreshed list of locations.

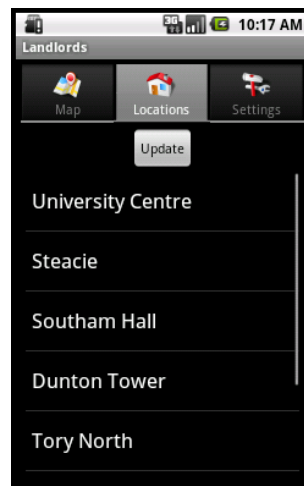


Figure 1: View of the Locations tab

The third tab is probably the most important tab; this is the map (see Figure 2: The Map view). When a player has GPS enabled on their device and are looking at the map tab, they will be provided with location updates. This is represented by the Android Robot in

the center of the screen, and reflects their most recent location. At this point, they have the ability to “Check In”. This, however, can be a gamble, and can result in any of the following scenarios:

- The player owns the location. In this case, they are returned a view of the locations information, and are provided with the opportunity to sell. Selling is not “instant”, however, and one must wait for another person to purchase the property before they will earn any money.
- If a location exists but is not currently owned by anybody, then it may be purchased for a set amount of money. Note that this is also the case if the owner has decided to sell this property.
- If another player owns the location and it is NOT for sale, then the player checking in will be charged rent. The amount of rent is determined when the property is created, and may not be changed afterwards.
- Finally, if the location does not currently exist, and the server has enabled location creation, then the user may create it. They can determine its name, the amount of rent to charge, and radius. The radius represents a circle around the specified latitude/longitude, and anybody checking in with coordinates inside that radius will be charged rent. It should be noted that the cost of the location is relative to the size of the circle and amount of rent charged, given by this equation, where C is a constant:

$$value = C \times rent \times \pi r^2$$

where C is a constant.

Although the location tracer does work and can easily be demonstrated, it is not currently used to define locations relative to the user. The reasons for this will be discussed later in this report.

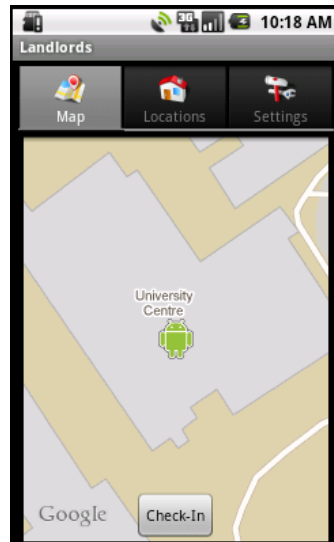


Figure 2: The Map view

Although there is technically no way to “win” this game, the goal is to earn as much money as you can. You amass your money by first purchasing properties, then charging rent to someone when they happen to check in to your property. This is intended to be a casual game, which means the player does not always need to be playing or their account to be making progress. For example, if they have triggered a property to be sold, and then signed out, anybody who comes along to the property and checks in will be given the option to buy. These transactions occur whether or not the user who owns the property is signed in. Therefore, it is not essential for a user to stay signed in and playing constantly; only at their leisure.

3. Methodology

No software can expect to be easily deployable and maintainable if it is not well structured and organized. In this section we will discuss some of the structure and code that is contained in the software.

a. Server

A quality server is essential to any distributed system. The server must handle a multitude of requests from users who wish to retrieve or update the material that it provides. Early in the development of the server, it was realized that the majority of the game is just a reflection on the state of the locations and users within the system. As such, communication between a client and the server is based almost entirely on acquiring a resource, or posting to one. This made the project an obvious candidate for the RESTful style of software architecture. REST stands for Representational State Transfer and was introduced by Roy Fielding in 2000.⁴ Since the internet is based on resource representation, it seems like a logical choice for a distributed application.

The server for this project is built using the Restlet framework which can be downloaded freely from their website. This provides access to the RESTful style via a plugin for Eclipse. Restlet was incredibly easy to install and setup, not to mention implement. Our server registers several resources such as locations or users to the root address. When they are requested by a client, the server triggers those resources to execute whatever code they may require, and then return data to the server. In this manner we were able to provide the ability to login, check in, purchase, sell, and get location lists upon request of a client.

⁴ http://en.wikipedia.org/wiki/Representational_State_Transfer

b. Client

However easy developing with the Restlet framework may have been, developing for the Android operating system was not as simple. The first problem is setting up the Android SDK. You must first install and setup the SDK and a variety of packages to get everything working. Once the SDK is installed, an Android Emulator can be created. This is one of the most useful tools when programming for the Android. This provides the developer with nearly full control of a mobile device from the comfort of their computer. Almost all of the functionality that is provided on the device is also available on the emulator. Even though there is no direct GPS in the emulator, it can be simulated in a variety of ways such as injecting locations in through the DDMS.

The client follows a RESTful approach to communication. The client follows a Service API in order to decouple functionality (see Figure 3: Service API).⁵

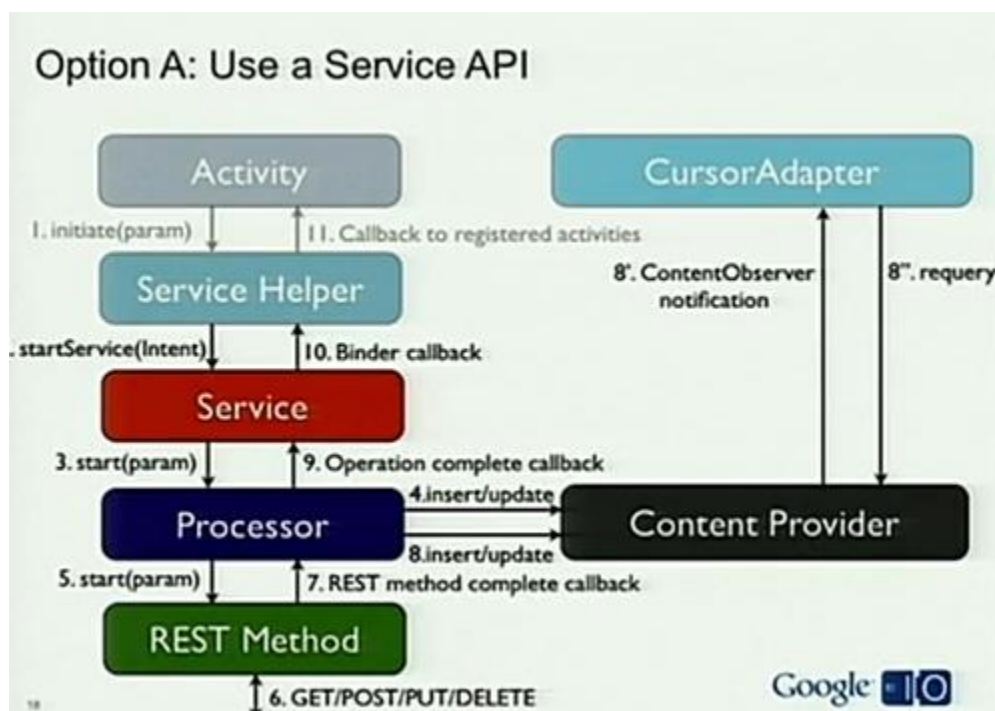


Figure 3: Service API

⁵ Google IO session with Virgil Dobjanschi, <http://www.youtube.com/watch?v=xHXn3Kg2IQE>

Essentially, whenever the user wishes to accomplish something, such as a button press to register a login, the Activity creates a ServiceHelper. It provides the ServiceHelper a URI for which to retrieve a resource, and then initiates the helper with these values, and registers a callback to the current Activity. The ServiceHelper then runs in the background to send and receive the data. Services are incredibly useful, as they provide our application a method to tell the Android device “I’m still alive!” As such, when the device wishes to shut something down to provide memory to another Activity, it will likely not choose our running Service. As such, Services can be a way to keep our application from being killed by the OS.

Another important part of our client is the LocationListener. Whenever we require a GPS (or from a different provider) location, we can register our listener to a LocationManager. Whenever GPS updates come in, our listener will be triggered and we can handle them in a meaningful way (such as updating our map). With this functionality, we can also unregister our listener when we no longer need location updates, so that we do not waste the battery by using the GPS.

One design choice that I chose was not to rely heavily on the strings.xml resource. This resource is incredibly useful for keeping strings organized in our application, and even provides a method of translating our strings depending on the set language (assuming we have multiple languages defined in our strings). However, with the scope of academic project, it is not intended to ever require translation to another language. Secondly, I decided to use a common class called “LandlordsConstants”. Since this class only provides final, static variables, both our server and client can reference it to make sure the request they are making is the correct one, and this provides a sense of security from requests breaking due to a misspelled or changed string on one side or the other.

c. Location Tracer

Probably the most interesting utility in this project, the location tracer gives us a method of tracing the outlines around buildings in Google Maps. This provides us with a method of saying “I’m at this location, what is there?” The location tracer can then call up the specific static map for that location, and decide if there is a building there or not. If there is, the location tracer does a rough approximation of the outline, and returns the polygon formed by the vertices.

The most important tool that aided me in accomplishing this was Google Maps. With their Static Maps API, you can just call the URI of the map and it will return it to the requester.⁶ Plus, you can customize the map via the query string. For example, the following portion of the URL will disable all features on the map, allowing us to only display ones that we need.

`&style=feature:all|element:all%7Cvisibility:off`

This functionality can also be used to change colour, zoom, and a variety of other factors.

The algorithm used to achieve the outlining is a modified version of the Lang Simplification algorithm.⁷ The algorithm has a “look ahead” value which can be used to set how many vertices ahead we want to start looking. For example, if “look ahead” is 8, we look 8 vertices forward and work backward. Then we analyze the projective distance from the in-between vertices to the line (from vertex 1 to vertex 8 in this example). If the distance is greater than the “tolerance” value, then this vertex is too far to “simplify”. We then look at the line from vertex 1 to vertex 7, and repeat the process until all the tolerances are simplified, at which point the in-between vertices are removed, and our algorithm moves on to the next line segment. This repeats until the outline stops, or it wraps back around to the initial vertex.

⁶ <http://code.google.com/apis/maps/documentation/staticmaps/#StyledMapFeatures>

⁷ <http://www.geom.unimelb.edu.au/gisweb/LGmodule/LGLangVisualisation.htm>

Since the above algorithm assumes that we already know all the vertices, it needs a small adjustment to work for our purposes. Since we are analyzing an image from Google, we first must represent what IS a building and what is not. To do this, we query the static map to return ONLY a map with a regular background and bright green buildings. This lets us analyze at a per-pixel level to determine whether or not the current location is a building or not. Once we have analyzed all our pixels and stored them in a boolean map (2D array of 0's and 1's which represent an empty pixel and an occupied pixel, respectively), we look at our user's current location. If the pixel they are on is empty, then we say "you're not in a building." However, if it is in a building, we move all the way to the right until we hit a pixel where there is empty space. Then we check all neighbours to see if they are empty or not. If a neighbor is occupied yet it has no empty neighbours, then we determine that it is an "inner vertex" and we do not touch use it.

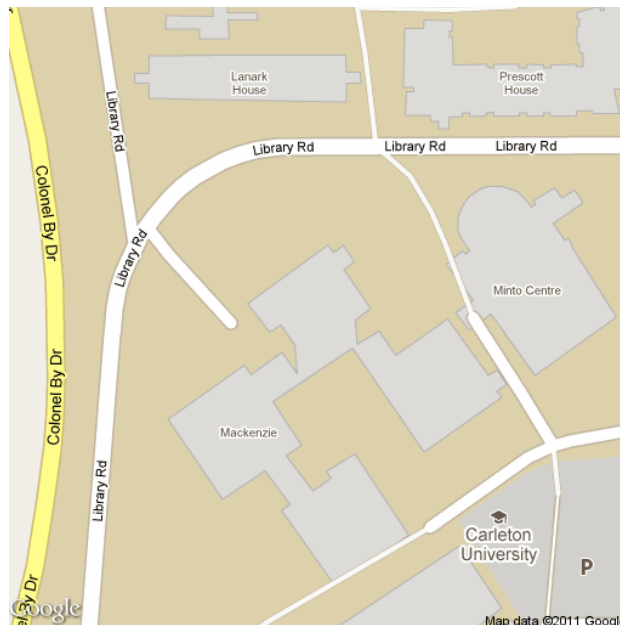


Figure 4: Typical Google map result

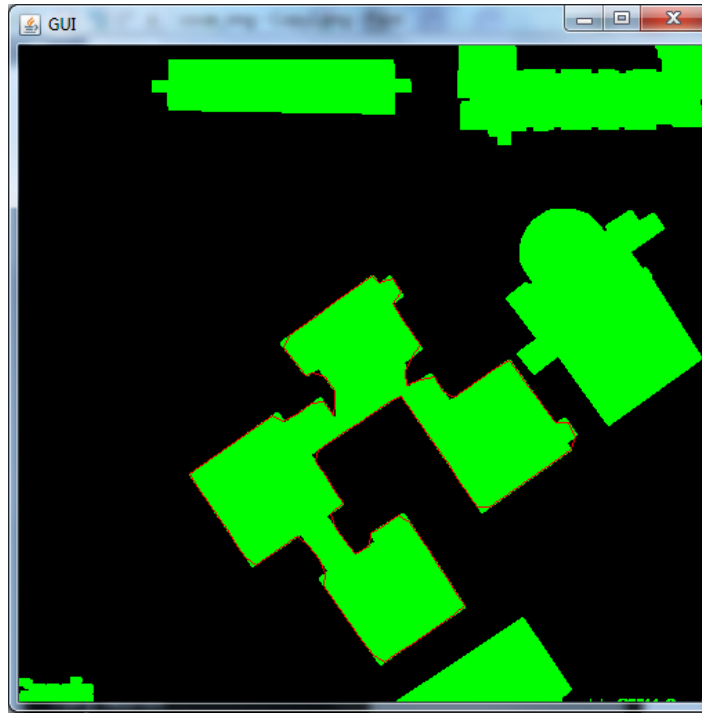


Figure 5: Map from Figure 4 after being read and traced

The importance of a pixel is determined by whether or not it is in the same wall as the previous pixel. This is especially important at corners, where choosing the wrong pixel could cause pixels to become unreachable (see Figure 6: Pixel exclusion). Therefore, our algorithm decides that since pixel C is a wall with the current pixel A, we choose it before we choose pixel B.

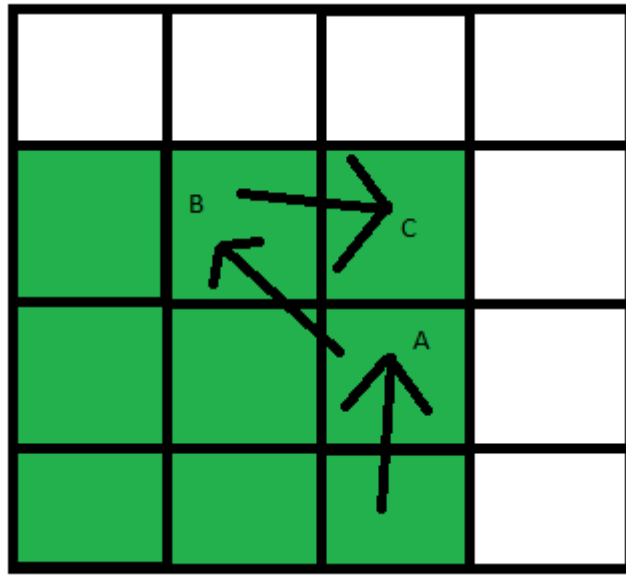


Figure 6: Pixel exclusion

4. Results

The result of this project is that we have gained some insight into the development of a distributed Android application. I have provided the methodology for the software, as well as why I made some of the decisions the way I did. In the end, we have a working Android application with a back-end server that is runnable on any Java ready computer.

Although there is not a considerable contribution to the field of Computer Science, it does provide a framework for later development of similar applications. Much of the code is there behind the scenes and readily accessible depending on the specific goal we wish to achieve.

Furthermore, this project has demonstrated a working algorithm for removing vertices from a polygon, and has been modified and extended to be used as an algorithm for tracing and providing rough outlines of a shapes made from the pixels of an image. This contribution to Computer Science may help provide a tutorial for tracing to people who are new to the concept, and may even be modified and optimized to be even faster and more efficient.

5. References

- [1] <http://www.cbsnews.com/stories/2010/02/15/business/main6209772.shtml>
- [2] <http://www.census.gov/ipc/www/idb/worldpopgraph.php>
- [3] <http://www.restlet.org/>
- [4] http://en.wikipedia.org/wiki/Representational_State_Transfer
- [5] Google IO session with Virgil Dobjanschi,
<http://www.youtube.com/watch?v=xHXn3Kg2IQE>
- [6] <http://code.google.com/apis/maps/documentation/staticmaps/#StyledMapFeatures>

6. Appendices

a. CD Content

Included on the CD is the full source code required to build and run. This will be found in the “Landlords\LandlordsProject\Source” directory.

A jar file for easily running the server, as well as a Debug and Release apk file for installing the client application on the Android Emulator and Android Device, respectfully. These are contained in the “Landlords\LandlordsProject\Release” directory. It is required that for them to be run, the entire folder.

This project report will also be contained on the CD under the “Landlords\Report” directory.

A user manual with information regarding how to run the various projects will be located in the “Landlords\Manual” directory.

The “cover_img.png” as well as several other image files are located in the “Report”.

Two batch files are located in the CD’s main directory. These represent methods of running the LocationTracer without running everything else. It is required that they stay relative to their path, therefore it is encouraged that the entire “Landlords” project be copied from the CD kept together.

b. Instructions

For instructions on how to run or setup the project, please see the Manual located in the |Landlords\Manual” directory of the CD.